Le but de ce TP est de se familiariser avec les concepts de processus sous le système GNU/Linux en ligne de commande et à partir de programmes simples. Pour chaque exécution du programme, vous noterez soigneusement les résultats affichés afin de pouvoir comparer les différentes exécutions.

1. Lister des processus

a. Affichage des *processus*

Proposez une commande permettant d'afficher tous les processus de tous les utilisateurs sous cette forme.

```
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.2  0.2 159884  9040 ?        Ss   10:51   0:02 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    10:51   0:00 [kthreadd]
root           4  0.0  0.0      0     0 ?        I<   10:51   0:00 [kworker/0:0H]
root           6  0.0  0.0      0     0 ?        I<   10:51   0:00 [mm_percpu_wq]
root           7  0.0  0.0      0     0 ?        S    10:51   0:00 [ksoftirqd/0]
root           8  0.0  0.0      0     0 ?        I    10:51   0:00 [rcu_sched]
root           9  0.0  0.0      0     0 ?        I    10:51   0:00 [rcu_bh]
root          10  0.0  0.0      0     0 ?        S    10:51   0:00 [migration/0]
root          11  0.0  0.0      0     0 ?        S    10:51   0:00 [watchdog/0]
root          12  0.0  0.0      0     0 ?        S    10:51   0:00 [cpuhp/0]
root          13  0.0  0.0      0     0 ?        S    10:51   0:00 [kdevtmpfs]
root          14  0.0  0.0      0     0 ?        I<   10:51   0:00 [netns]
root          15  0.0  0.0      0     0 ?        S    10:51   0:00 [rcu_tasks_kthre]
```

b. Hiérarchie des processus

Quelle commande utiliser pour afficher l'arbre des processus ?

```
?─┬─ModemManager───2*[{ModemManager}]
  ├─NetworkManager─┬─dhclient
  │                └─2*[{NetworkManager}]
  ├─accounts-daemon───2*[{accounts-daemon}]
  ├─acpid
  ├─avahi-daemon───avahi-daemon
  ├─boltd───2*[{boltd}]
  ├─colord───2*[{colord}]
  ├─cron
  ├─cups-browsed───2*[{cups-browsed}]
  ├─cupsd───2*[dbus]
  ├─dbus-daemon
  ├─fwupd───4*[{fwupd}]
  ├─gdm3─┬─gdm-session-wor─┬─gdm-x-session─┬─Xorg───{Xorg}
  │      │                 │               └─gnome-session-b─┬─deja-dup-monito───3*
  │      │                 │                                 ├─gnome-shell─┬─ibus-d
  │      │                 │                                 │             └─9*[{gn
  │      │                 │                                 ├─gnome-software───3*[
  │      │                 │                                 ├─gsd-a11y-settin───3*
```

Quel est le nom du premier processus ? Quel est son PID ?

c. Tuer un processus

Quelle est la commande pour tuer un processus ?

d. Etat des processus

Quels sont les différents états d'un processus ? Les décrire.

2. Programmation simple de processus

a. Identification du père et du fils

Écrivez un programme C s'exécutant sous la forme de deux processus, père et fils. Utilisez pour cela la fonction :

```
pid_t fork(void)
```

Elle crée une copie conforme du processus en dupliquant toutes les entrées du processus en cours, y compris les descripteurs de fichiers ouverts. Le processus appelant la fonction *fork()* devient dès lors le processus père, et le processus nouvellement créé, le processus fils. Après l'appel, on distingue les deux processus en testant la valeur de retour. Elle contient le PID du fils pour le processus père, et 0 pour le fils.

Le père doit afficher l'identité (PID) du fils créé.

Le fils doit être composé d'une boucle infinie dans laquelle il affiche son PID toutes les 5 secondes. Au niveau de l'exécution, vous pourrez observer le fait que le processus fils continue alors que le père est terminé.

Vous utiliserez donc également les fonctions :

```
pid_t get_pid(void)
```

qui retourne le PID du processus courant et :

```
void sleep(unsigned int time)
```

qui endort le processus pendant un nombre de secondes passé en paramètre, ou jusqu'à ce qu'un signal non-ignoré soit reçu.

Toutes ces fonctions sont définies dans le fichier /usr/include/unistd.h, le type *pid_t* est quant à lui défini dans /usr/include/sys/types.h.

b. Zombies !

Ajoutez ensuite une boucle infinie dans le père. A l'exécution, observez tous les processus en cours. Puis tuez le processus père. Qu'observez-vous ?

Après avoir assaini la liste des processus, relancez une exécution en tuant le processus fils en premier. Qu'observez-vous ?

Remarque importante : les résultats obtenus dépendent du système d'exploitation (Linux, OSX, …) et du shell utilisé (csh, bash, …). Comparez vos résultats !

Ecrire vos réponses en commentaire dans le programme en précisant votre environnement (OS et shell).